User Manual For Real-Time Sensor Data Streaming

Harmit Khimani & Nishank Kansara

B.Tech. Mini Project - Under Prof. Tapas Kumar Maiti

Contents

1 Prerequisites

Ensure you have the following installed and configured:

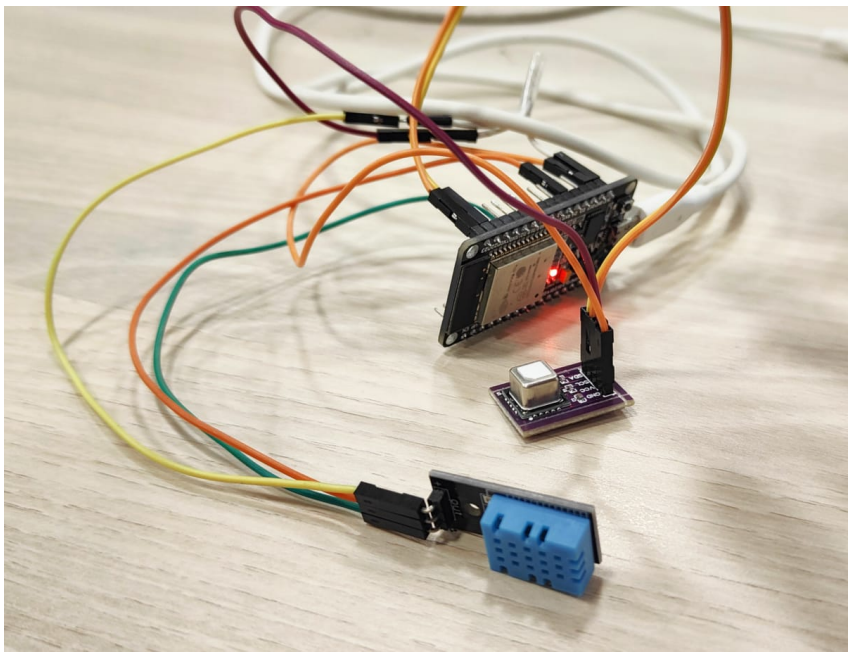
- **Java 11+**
- **Apache Kafka 3.9.0 (Scala 2.13)**
- **ESP32 toolchain** (Arduino IDE or PlatformIO)
- Network connectivity: ESP32, Kafka server, and Spring Boot host on the same LAN or Wi-Fi.

2 Hardware Setup & Wiring

Connect sensors directly to the ESP32 via USB power.

Sensor	ESP32 Pin	Notes
DHT11	GPIO 4	Data line (10 KΩ pull-up)
SCD40	SDA → GPIO 22, SCL → GPIO 23	I^2C bus lines
VCC	3V3	3.3 V from ESP32
GND	GND	Common ground

Table 1: Sensor to ESP32 Wiring



3 ESP32 MQTT Publisher Code

3.1 Install Libraries

Add these libraries to your Arduino IDE:

- **PubSubClient** for MQTT
- **DHT sensor** library
- **SCD4x** library for CO_2 sensing
- **Wire** for I^2C communication

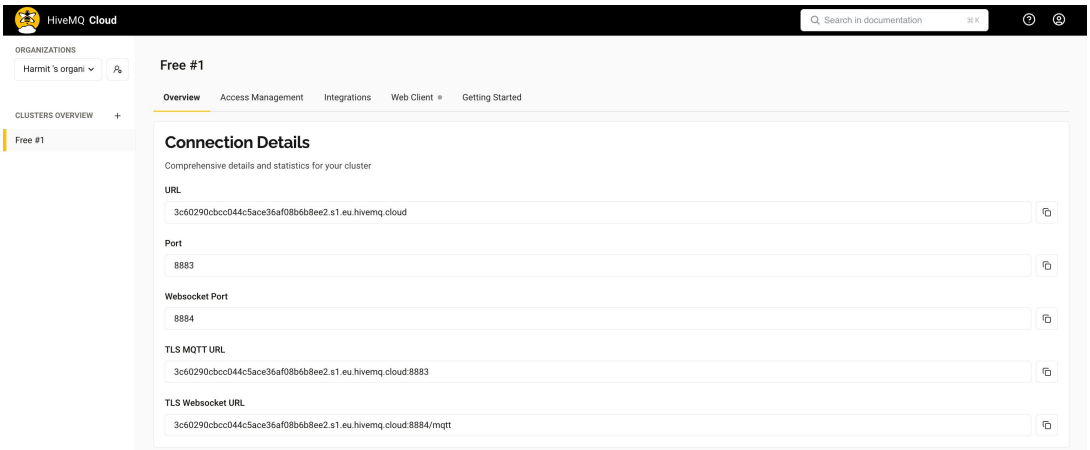
3.2 Embed MQTT Credentials

```
//  W i Fi
const char* ssid     = "<YOUR_SSID>";
const char* password = "<YOUR_WIFI_PASSWORD>";

// HiveMQ Cloud
const char* mqtt_server   = "<YOUR_MQTT_SERVER_URL>";
const int   mqtt_port     = 8883;
const char* mqtt_username = "<USERNAME>";
const char* mqtt_password = "<PASSWORD>";
const char* topic         = "<TOPIC>";
```

3.3 Obtain HiveMQ Cloud Details

1. Sign in at `https://console.hivemq.cloud`.
2. Create a free cluster.
3. Copy the **Cluster Endpoint** and paste it into the `mqtt_server` configuration field.
4. Retrieve or generate your **Credentials** for a secure connection.



4 Kafka Setup

4.1 Disable Firewall

Open port `9092`:

Windows: `netsh advfirewall firewall add rule name="KafkaPort" dir=in action=allow protocol=TCP localport=9092`

macOS (Terminal): `sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate off`

4.2 Download Kafka 3.9.0

1. Visit `https://kafka.apache.org/downloads`.
2. Select Scala 2.13 + Kafka 3.9.0.
3. Extract the archive.

4.3 Configure Kafka Listeners

In `config/server.properties`:

```
listeners=PLAINTEXT://0.0.0.0:9092
advertised.listeners=PLAINTEXT://<YOUR_HOST_IP>:9092
```

4.4 Start ZooKeeper

```
cd kafka_2.13-3.9.0
bin/zookeeper-server-start.sh config/zookeeper.properties
```

4.5 Start Kafka Server

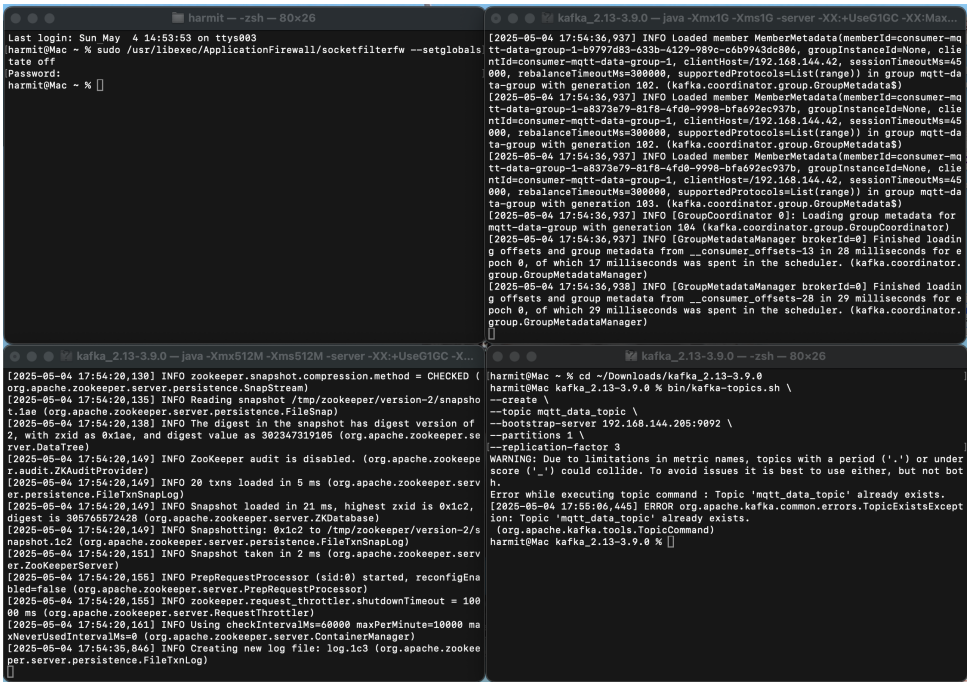
```
cd kafka_2.13-3.9.0
bin/kafka-server-start.sh config/server.properties
```

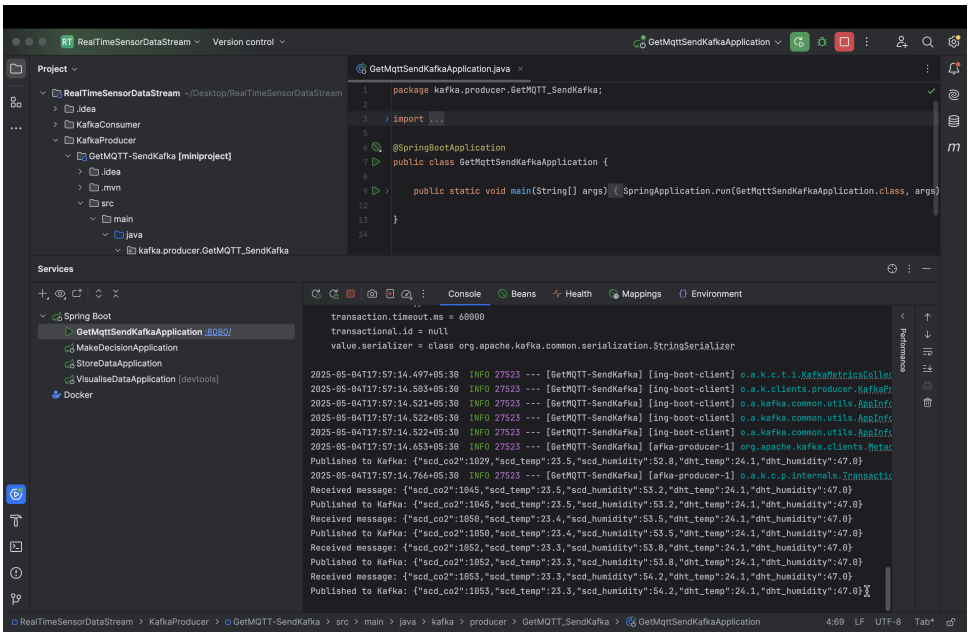
4.6 Create Verify Topic

```
# Create topic
bin/kafka-topics.sh --create \
  --topic esp32-mqtt-topic \
  --bootstrap-server <BROKER_IP>:9092 \
  --partitions 1 --replication-factor 3

# List topics
echo "Available topics:" && \
bin/kafka-topics.sh --list --bootstrap-server localhost:9092

# Test consumption
bin/kafka-console-consumer.sh \
  --bootstrap-server <BROKER_IP>:9092 \
  --topic esp32-mqtt-topic \
  --from-beginning
```


harmit — -zsh — 80×26
Last login: Sun May 4 14:53:53 on ttys003
harmit@Mac ~ % sudo /usr/libexec/ApplicationFirewall/socketfilterfw --setglobalstate off
Password:
harmit@Mac ~ %
kafka_2.13-3.9.0 — java -Xmx1G -Xms1G -server -XX:+UseG1GC -XX:Max...
[2025-05-04 17:54:36,937] INFO Loaded member MemberMetadata(memberId=consumer-mqtt-data-group-1-b9797d83-633b-4129-989c-c6b994 3dc806, groupInstanceId=None, clientId=consumer-mqtt-data-group-1, clientHost=/192.168.144.42, sessionTimeoutMs=45000, rebalanceTimeoutMs=300000, supportedProtocols=List(range)) in group mqtt-data-group with generation 102. (kafka.coordinator.group.GroupMetadata$)
[2025-05-04 17:54:36,937] INFO Loaded member MemberMetadata(memberId=consumer-mqtt-data-group-1-a8373e79-81f8-4fd0-9998-bfa692ec937b, groupInstanceId=None, clientId=consumer-mqtt-data-group-1, clientHost=/192.168.144.42, sessionTimeoutMs=45000, rebalanceTimeoutMs=300000, supportedProtocols=List(range)) in group mqtt-data-group with generation 102. (kafka.coordinator.group.GroupMetadata$)
[2025-05-04 17:54:36,937] INFO Loaded member MemberMetadata(memberId=consumer-mqtt-data-group-1-a8373e79-81f8-4fd0-9998-bfa692ec937b, groupInstanceId=None, clientId=consumer-mqtt-data-group-1, clientHost=/192.168.144.42, sessionTimeoutMs=45000, rebalanceTimeoutMs=300000, supportedProtocols=List(range)) in group mqtt-data-group with generation 103. (kafka.coordinator.group.GroupMetadata$)
[2025-05-04 17:54:36,937] INFO [GroupCoordinator 0]: Loading group metadata for mqtt-data-group with generation 104 (kafka.coordinator.group.GroupCoordinator)
[2025-05-04 17:54:36,937] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-13 in 28 milliseconds for epoch 0, of which 17 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
[2025-05-04 17:54:36,938] INFO [GroupMetadataManager brokerId=0] Finished loading offsets and group metadata from __consumer_offsets-28 in 29 milliseconds for epoch 0, of which 29 milliseconds was spent in the scheduler. (kafka.coordinator.group.GroupMetadataManager)
kafka_2.13-3.9.0 — java -Xmx512M -Xms512M -server -XX:+UseG1GC -X...
[2025-05-04 17:54:20,130] INFO zookeeper.snapshot.compression.method = CHECKED (org.apache.zookeeper.server.persistence.SnapStream)
[2025-05-04 17:54:20,135] INFO Reading snapshot /tmp/zookeeper/version-2/snapshot.1ae (org.apache.zookeeper.server.persistence.FileSnap)
[2025-05-04 17:54:20,138] INFO The digest in the snapshot has digest version of 2, with zxid as 0x1ae, and digest value as 302347319105 (org.apache.zookeeper.server.DataTree)
[2025-05-04 17:54:20,149] INFO ZooKeeper audit is disabled. (org.apache.zookeeper.audit.ZKAuditProvider)
[2025-05-04 17:54:20,149] INFO 20 txns loaded in 5 ms (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2025-05-04 17:54:20,149] INFO Snapshot loaded in 21 ms, highest zxid is 0x1c2, digest is 305765572428 (org.apache.zookeeper.server.ZKDatabase)
[2025-05-04 17:54:20,149] INFO Snapshotting: 0x1c2 to /tmp/zookeeper/version-2/snapshot.1c2 (org.apache.zookeeper.server.persistence.FileTxnSnapLog)
[2025-05-04 17:54:20,151] INFO Snapshot taken in 2 ms (org.apache.zookeeper.server.ZooKeeperServer)
[2025-05-04 17:54:20,155] INFO PrepRequestProcessor (sid:0) started, reconfigEnabled=false (org.apache.zookeeper.server.PrepRequestProcessor)
[2025-05-04 17:54:20,155] INFO zookeeper.request_throttler.shutdownTimeout = 10000 ms (org.apache.zookeeper.server.RequestThrottler)
[2025-05-04 17:54:20,161] INFO Using checkIntervalMs=60000 maxPerMinute=10000 maxNeverUsedIntervalMs=0 (org.apache.zookeeper.server.ContainerManager)
[2025-05-04 17:54:35,846] INFO Creating new log file: log.1c3 (org.apache.zookeeper.server.persistence.FileTxnLog)
kafka_2.13-3.9.0 — -zsh — 80×26
harmit@Mac ~ % cd ~/Downloads/kafka_2.13-3.9.0
harmit@Mac kafka_2.13-3.9.0 % bin/kafka-topics.sh \
--create \
--topic mqtt_data_topic \
--bootstrap-server 192.168.144.205:9092 \
--partitions 1 \
--replication-factor 3
WARNING: Due to limitations in metric names, topics with a period ('.') or underscore ('_') could collide. To avoid issues it is best to use either, but not both.
Error while executing topic command : Topic 'mqtt_data_topic' already exists.
[2025-05-04 17:55:06,445] ERROR org.apache.kafka.common.errors.TopicExistsException: Topic 'mqtt_data_topic' already exists.
(org.apache.kafka.tools.TopicCommand)
harmit@Mac kafka_2.13-3.9.0 %


RealTimeSensorDataStream
Version control
GetMqttSendKafkaApplication
Project
RealTimeSensorDataStream ~/Desktop/RealTimeSensorDataStream
.idea
KafkaConsumer
KafkaProducer
GetMQTT-SendKafka [miniproject]
.idea
.mvn
src
main
java
kafka.producer.GetMQTT_SendKafka
GetMqttSendKafkaApplication.java
package kafka.producer.GetMQTT_SendKafka;
import ...
@SpringBootApplication
public class GetMqttSendKafkaApplication {
public static void main(String[] args) { SpringApplication.run(GetMqttSendKafkaApplication.class, args)
}
Services
Spring Boot
GetMqttSendKafkaApplication :8080/
MakeDecisionApplication
StoreDataApplication
VisualiseDataApplication [devtools]
Docker
Console
Beans
Health
Mappings
Environment
transaction.timeout.ms = 60000
transactional.id = null
value.serializer = class org.apache.kafka.common.serialization.StringSerializer
2025-05-04T17:57:14.497+05:30 INFO 27523 --- [GetMQTT-SendKafka] [ing-boot-client] o.a.k.c.t.i.KafkaMetricsCollec
2025-05-04T17:57:14.503+05:30 INFO 27523 --- [GetMQTT-SendKafka] [ing-boot-client] o.a.k.clients.producer.KafkaPr
2025-05-04T17:57:14.521+05:30 INFO 27523 --- [GetMQTT-SendKafka] [ing-boot-client] o.a.kafka.common.utils.AppInfo
2025-05-04T17:57:14.522+05:30 INFO 27523 --- [GetMQTT-SendKafka] [ing-boot-client] o.a.kafka.common.utils.AppInfo
2025-05-04T17:57:14.522+05:30 INFO 27523 --- [GetMQTT-SendKafka] [ing-boot-client] o.a.kafka.common.utils.AppInfo
2025-05-04T17:57:14.653+05:30 INFO 27523 --- [GetMQTT-SendKafka] [afka-producer-1] org.apache.kafka.clients.Metad
Published to Kafka: {"scd_co2":1029,"scd_temp":23.5,"scd_humidity":52.8,"dht_temp":24.1,"dht_humidity":47.0}
2025-05-04T17:57:14.766+05:30 INFO 27523 --- [GetMQTT-SendKafka] [afka-producer-1] o.a.k.c.p.internals.TransactionM
Received message: {"scd_co2":1045,"scd_temp":23.5,"scd_humidity":53.2,"dht_temp":24.1,"dht_humidity":47.0}
Published to Kafka: {"scd_co2":1045,"scd_temp":23.5,"scd_humidity":53.2,"dht_temp":24.1,"dht_humidity":47.0}
Received message: {"scd_co2":1050,"scd_temp":23.4,"scd_humidity":53.5,"dht_temp":24.1,"dht_humidity":47.0}
Published to Kafka: {"scd_co2":1050,"scd_temp":23.4,"scd_humidity":53.5,"dht_temp":24.1,"dht_humidity":47.0}
Received message: {"scd_co2":1052,"scd_temp":23.3,"scd_humidity":53.8,"dht_temp":24.1,"dht_humidity":47.0}
Published to Kafka: {"scd_co2":1052,"scd_temp":23.3,"scd_humidity":53.8,"dht_temp":24.1,"dht_humidity":47.0}
Received message: {"scd_co2":1053,"scd_temp":23.3,"scd_humidity":54.2,"dht_temp":24.1,"dht_humidity":47.0}
Published to Kafka: {"scd_co2":1053,"scd_temp":23.3,"scd_humidity":54.2,"dht_temp":24.1,"dht_humidity":47.0}
RealTimeSensorDataStream > KafkaProducer > GetMQTT-SendKafka > src > main > java > kafka > producer > GetMQTT_SendKafka > GetMqttSendKafkaApplication
4:69 LF UTF-8 Tab*

5 Data Storage, Visualization & Smart Feedback

5.1 PostgreSQL (PGAdmin)

Store historical sensor data:

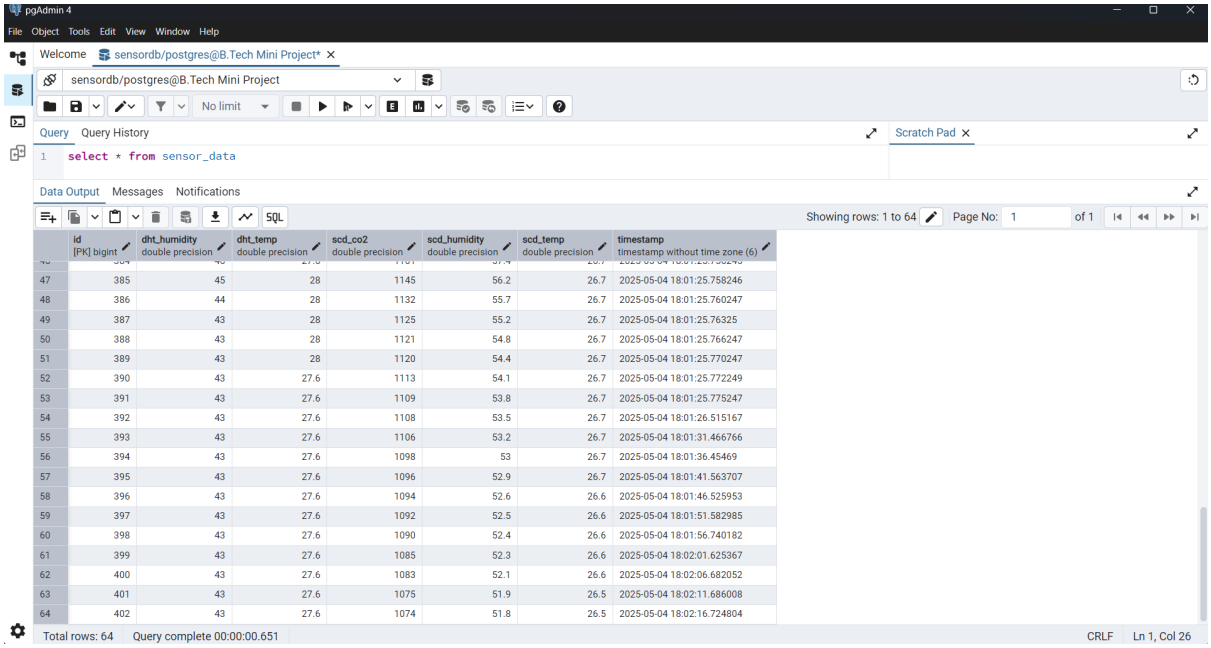
1. Run the SaveData application to store the data via API
2. Launch PGAdmin, connect to `localhost:5432`.
3. Create a database named `sensor_db`.
4. run the SQL Query

```
SELECT * from sensor_data
```

5. to delete the data fromt he database

```
DELETE from sensor_data
```

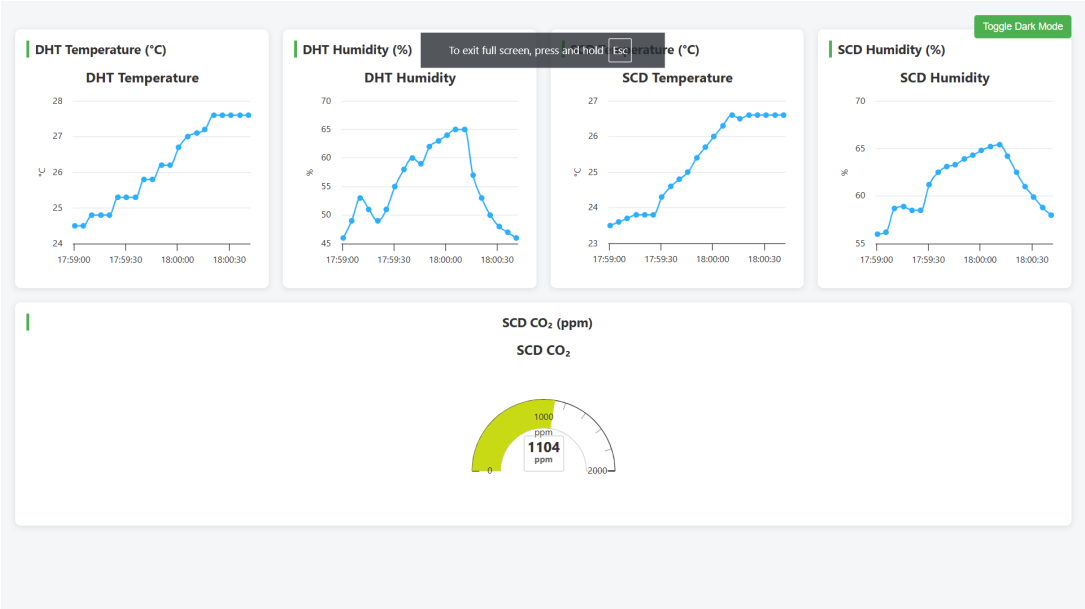
Spring Data Service This service receives sensor data via Kafka and stores it in PostgreSQL. Run the Spring Boot service on port `8082` to enable database storage.



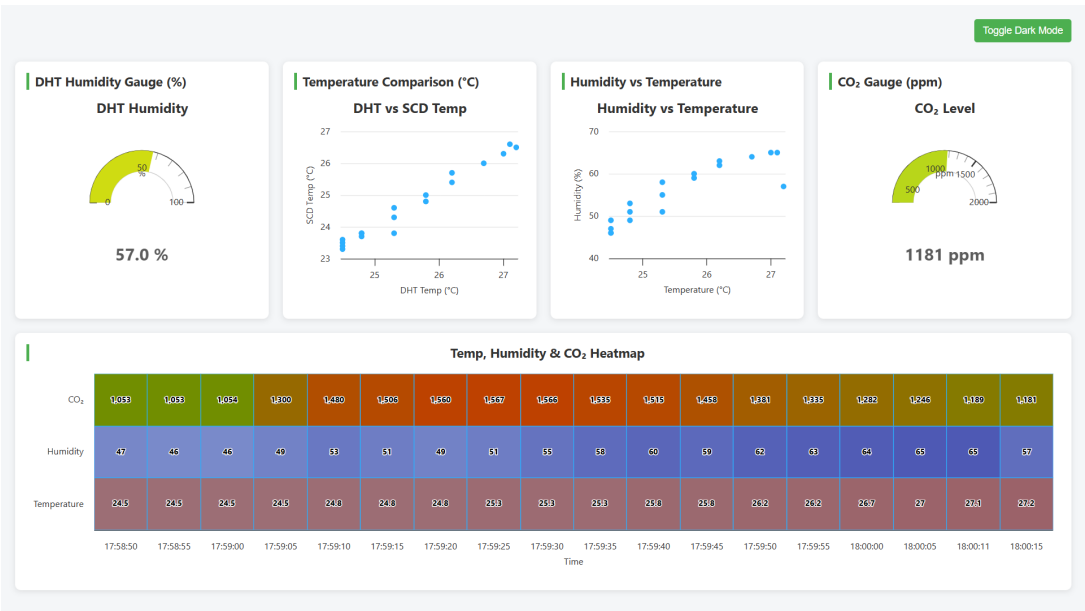
	id [PK] bigint	dht_humidity double precision	dht_temp double precision	scd_co2 double precision	scd_humidity double precision	scd_temp double precision	timestamp timestamp without time zone (6)
47	385	45	28	1145	56.2	26.7	2025-05-04 18:01:25.758246
48	386	44	28	1132	55.7	26.7	2025-05-04 18:01:25.760247
49	387	43	28	1125	55.2	26.7	2025-05-04 18:01:25.76325
50	388	43	28	1121	54.8	26.7	2025-05-04 18:01:25.766247
51	389	43	28	1120	54.4	26.7	2025-05-04 18:01:25.770247
52	390	43	27.6	1113	54.1	26.7	2025-05-04 18:01:25.772249
53	391	43	27.6	1109	53.8	26.7	2025-05-04 18:01:25.775247
54	392	43	27.6	1108	53.5	26.7	2025-05-04 18:01:26.515167
55	393	43	27.6	1106	53.2	26.7	2025-05-04 18:01:31.466766
56	394	43	27.6	1098	53	26.7	2025-05-04 18:01:36.45469
57	395	43	27.6	1096	52.9	26.7	2025-05-04 18:01:41.563707
58	396	43	27.6	1094	52.6	26.6	2025-05-04 18:01:46.525953
59	397	43	27.6	1092	52.5	26.6	2025-05-04 18:01:51.582985
60	398	43	27.6	1090	52.4	26.6	2025-05-04 18:01:56.740182
61	399	43	27.6	1085	52.3	26.6	2025-05-04 18:02:01.625367
62	400	43	27.6	1083	52.1	26.6	2025-05-04 18:02:06.682052
63	401	43	27.6	1075	51.9	26.5	2025-05-04 18:02:11.686008
64	402	43	27.6	1074	51.8	26.5	2025-05-04 18:02:16.724804

5.2 Web Visualization

- **CO_2, Temperature, and Humidity Line Charts:** `localhost:8081/basic-chart.html`
  - Line charts for DHT11 Humidity & Temperature, SCD40 Humidity & Temperature.
  - A real-time gauge for CO_2 concentration levels.



- **Advanced Comparative & Distribution Visualization:** `localhost:8081/advanced-chart.html`
  - A scatter chart comparing Temperature vs. Humidity.
  - CO_2 and Humidity gauges.
  - A scatter chart comparing temperature between the SCD40 & DHT11 sensors.
  - A heatmap of the distribution of CO_2, Humidity, and Temp values over time.

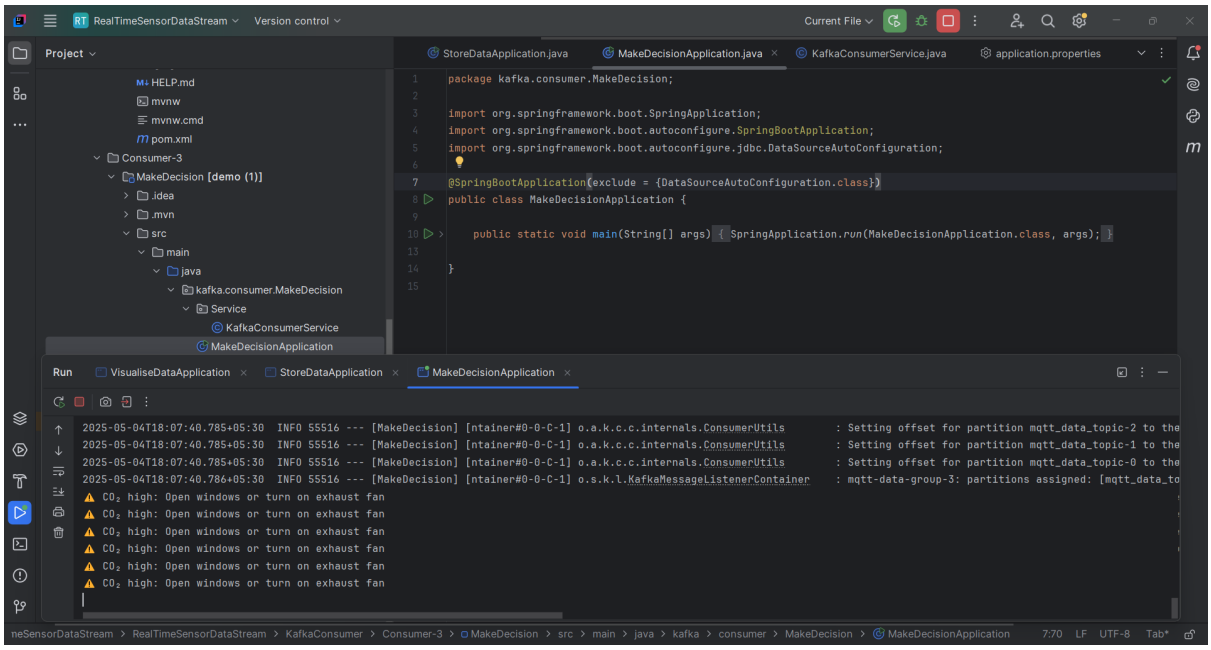


5.3 Real-Time Smart Feedback

Threshold-based suggestions:

Parameter	Threshold	Action
CO_2 Level	> 1200 ppm	Open windows or turn on exhaust fan
Temp	> 28 °C	Turn on fan or AC
Temp	> 20 °C	Use heater
Humidity	$> 35\%$	Use humidifier
Humidity	$> 65\%$	Use ventilation or dehumidifier

Table 2: Smart Feedback Rules



6 Additional Resources

- Kafka Downloads: `https://kafka.apache.org/downloads`
- Kafka Tutorial: `https://www.youtube.com/watch?v=tU_37niRh4U`

7 Troubleshooting

Symptom	Fix
No MQTT connection	Check ESP32 Wi-Fi HiveMQ credentials
No Kafka data	Ensure topic exists correct broker URL
Dashboard not loading	Verify Spring Boot service on port 8081
Firewall issues	Confirm port 9092 is open

Startup Sequence:

1. ESP32 publishes MQTT → Kafka
2. Start ZooKeeper
3. Start Kafka server
4. Run Kafka consumer console (test)
5. Launch Spring Boot apps (DB Dashboard)